SWE544
INTERNET PROGRAMMING PROJECT
TOMBALA

Final Report

Yasemin Alpay

12.01.2016

1. PREVIOUS WORK

In the design and requirements document of the project; structure, game-play and application protocol of the project is defined. There have been slight changes in the game-play and application protocol.

1.1 Game-Play

Following rules for the game are implemented like stated in the design and requirements project:

- *There will be several users that can connect to the server. Each user can either join a session, or create his/her own new session to start a game.*
- *Right after the game starts, server will draw new cards for every user in the session. There will be 3 rows in each card and every row consists of 5 numbers in the range of 1 to 90.*
- *After each user gets his/her card, server will pick numbers from 1 to 90 and sends this number to every user to check. Once a number is sent, it cannot be sent again.*
- *User will check the number that has been gotten from server and try to match this number with the numbers in his/her card. If there is a match, he/she will sign the number in the card and will send a signal to the server that he/she is ready for the next number. If there is not any match, he/she will send the signal without signing any number in the card.*
- *If a user is successful in matching every number in a row, he/she will call for "Çinko" and sends this signal to the server. Server will verify the "Çinko" and sends a signal if "çinko" is true or not. If it is true, this signal is sent to every user in the session with the message "First/Second Çinko by user xxx".*
- *If a user makes "Çinko" for every row, he/she can call for "Tombala/Bingo". This signal is verified again by the server. If it is successful, "Game is over, winner is user xxx" message is sent to every user in the session. Afterwards, session will be ended.*

1.1.1. Changes

Previous:

Once a user joins or starts a session, he/she will wait for other users to join the game. The game will start in 15 seconds if two users have been acquired at minimum. The maximum number of users that can join the game is five.

Now:

Number of users in a session is pre-defined with the variable `MAX_USER`. There is not a time out for the game to begin.

1.2. Application Protocol

1.2.1 Messages from Client to Server

Changes in the protocol have shown in Table 1. Additions have shown in **bold green**. Deletions have shown in **bold red.**

Request	Parameter	Response	Parameter	Definition
USR	nickname	HEL	nickname	User accepted
		REJ		Nickname exists
QUI		BYE		User exits
LSQ		LSA	session**:**,session2…	List sessions
LUQ	sessionname	LUA	nick1**:**,nick2…	List users in a session
		LNA		**User cannot list users before joining the session**
JOS	sessionname	JOK		Joins session
		FUL		**Session is full**
		JER		**Session not exists**
NES	sessionname	SOK		Creates new session
		SER		Session name exists
RDY		NEW	cardlist	Gets new card from server
NXT		NUM	random_number	Gets new number from server
CNK	cardlist,**cinko_count**	COK	**nick:nth cinko** **nickname** **cinko_row_list**	Sends user name who gets "Çinko"
		CER		Invalid "Çinko"
TOM	cardlist,**cinko_count**	TOK	nickname	Prints winner name
		TER		Invalid "Tombala"
FIN		END		Game over
Command		ERR		Wrong command
Command		ERL		User not logged in

Table 1: Application Protocol Client-Side Changes

1.2.2 Messages from Server to Client

Changes in the protocol have shown in Table 2. Additions have shown in **bold green**. Deletions have shown in **bold red.**

Request	Parameter	Response	Parameter	Definition
TIC		TOC		Connection test
JOK		RDY		User connects the session and ready for new game
SOK		RDY		User creates new session and ready for new game
NEW	cardlist	NXT		User waits for new number
NUM	random_number	NXT		User waits for new number
TOK	**nickname: "Tombala"** **nickname**	FIN		Session is closed

Table 2: Application Protocol Server-Side Changes

1.3. User Commands

There isn't any change in user commands.

All interactions between client and server will be within command-line.
User can enter specific commands to play the game:

/nick <nickname>	*: Changes nickname*
/quit	*: Quits server*
/list session	*: List sessions in the server*
/list user	*: List users in the session which user is playing in right now*
/join <session name>	*: Joins an existing session*
/new <session name>	*: Creates new session*
/next	*: Ready for next number*
/close <number>	*: Sign number in the card*
/çinko	*: Calls for "çinko"*
/tombala	*: Calls for "Tombala"*
/help	*: Prints all commands*

2. IMPLEMENTATION

2.1 Server Side

In the server side, there is one thread to read requests from clients. Requests are parsed in incoming_parser function of thread. Afterwards, the responses are sent from this thread in parser. If the response is going to be sent to all clients, it's sent in broadcastResponse function, otherwise it's sent from client's socket only.

In the server, there can be multiple sessions for users to play the games. User can create new session or list existing sessions and join one. There are 2 dictionary variables to hold the information about users and sessions (`user_info` and `session_info`).

`user_info:` Holds the information of users in a dictionary. Keys of this dictionary are socket objects that clients connect to server from, and values of this dictionary are the nick-session tuples.

Example:

```
user_info = {csoc1: {'nick': ‘john’, 'session': ‘pros’}, csoc2: {'nick': ‘watson’, 'session': ‘pros’} }
```

`session_info`: Holds all information about sessions including:

- Card lists that going to be pushed and popped to users when a new game begins. This list is converted to an OrderedDict in client-side. Because the order of the element is important to play the game. Otherwise a number's row can be changed.
- Random numbers that going to be sent to all clients to play the game.
- Past numbers that have been sent to all clients.
- User list of the session.

User list holds the following information:

- Socket objects as keys
- State and cinko_rows as values.
  Initial state when the game starts is ‘’. When the user checks his/her card and ready for new number, state is ‘1’. After new number is sent to all clients, state is ‘0’. New number is only going to be sent to clients if all the users are ready (state = ‘1’).
  cinko_rows is a list that holds user's çinko information. It is empty if user hasn't called for çinko yet. If he/she has a çinko, the row number of the çinko will be in this list.

Example:

```
Session_info = {'mysession': {'cards': [[86, 40, 20, 73, 49, 68, 87, 64, 2,
21, 28, 45, 25, 16, 18]], 'random_numbers': [77, 41, 11, 85, 63, 88, 4, 67,
27, 6, 64, 50, 86, 39, 9, 18, 38, 23, 76, 56, 52, 69, 12, 79, 30, 70, 37,
58, 31, 78, 35, 61, 72, 13, 3, 68, 15, 49, 66, 5, 51, 59, 8, 74, 57, 42,
17, 19, 62, 71, 84, 82, 33, 40, 87, 73, 83, 26, 47, 34, 36, 1, 53, 25, 54,
2, 32, 21, 46, 16, 60, 75, 44, 45, 89, 24, 10, 55, 80, 7, 65, 29, 14, 90],
'past_numbers' :[ 81, 22, 48, 28, 20, 43] 'users': {'socket1':
{'cinko_rows': [], 'state': '1'}, 'socket143': {'cinko_rows': [2, 1],
'state': ''}}}}
```

All the protocols are built on top of these 2 variables (user_info and session_info).

2.2 Client Side

There are 3 threads in the client side. One is for reading the server responses and printing them on the screen (ReadThread), one is for sending requests to the server (WriteThread) and one is for reading commands in the screen and push these commands to writeThread to send commands to the server (ScreenThread). There is a parser in the readThread and it is used for parsing server responses.

Card list holds the numbers of the card. It's printed on the screen in a special format each time a new number arrives or user closes a number. When user closes a number, it's checked with '*' sign. User can close a number only if the number is sent from the server and that number is in user's card. Numbers that server sends are kept in a list in the client-side. Therefore, user can close a number anytime he/she wants if the conditions met.

Each time user wants to check çinko or tombala, client sends CNK (çinko) or TOM (tombala) request to the server. If çinko(s) are verified, COK response with the cinko_row_list is sent back to client. cinko_row_list holds the row number of çinko(s) if there's any, so the length of this number will result çinko count or tombala if çinko count is 3.

3. ASSUMPTIONS AND BUGS

- User should login first to play the game.
- User should create a new session. Other clients can join this session.
- There are problems when sending responses to multiple clients.
- User should be checked with TIC-TOC signals to ensure that he/she is still on the session. Otherwise he/she should be removed.
- Remove operations should be efficient when the game is over.